WHITE PAPER

Getting started with Continuous Integration in software development

- Amruta Kumbhar, Madhavi Shailaja & Ravi Shankar Anupindi

Infosys®

Introduction

DevOps culture is gaining rapid momentum in the IT industry as it enables business to adopt agile software delivery methodologies like Continuous Integration (henceforth referred to as CI) & Continuous Delivery (henceforth referred to as CD) .These methodologies enable quicker issue resolution, instant feedback loops, improved software quality and cost saving to meet the ever-increasing demand to deliver better software faster. It would not be wrong to say that CI-CD practices will soon become the de-facto software delivery standards across the industry.

Gartner says "By 2016, DevOps will evolve from a niche to a mainstream strategy employed by 25 Percent of global 2000 organizations"

Though both these methodologies (CI and CD) complement each other, CI is the pre-requisite phase for enabling CD as the latter is built on top of the former. The primary goal of CI is not only to enable build automation through continuous test and quality checks but also to provide project insights through reports and dashboards. Since this phase is all about tools, it imposes various integration challenges. Having a good knowledge of the tools involved, their integration aspects and the best practices to follow, will definitely enable their smoother adoption and rollout across enterprises looking to adopt CI practices.

This article compiles insights gained from our practical experiences across various CI enablement programs which we have been associated with.

Some of these experiences might vary depending on the choice of tools, infrastructure setup, organization policies and project requirements. In spite of the differences, we hope, this article will act as a helping guide to all those planning for CI setup in their projects.

This article is organized into different sections as below –

Section 1 shows the CI setup which we have been using.

Section 2 talks about the various tools being used to achieve CI.

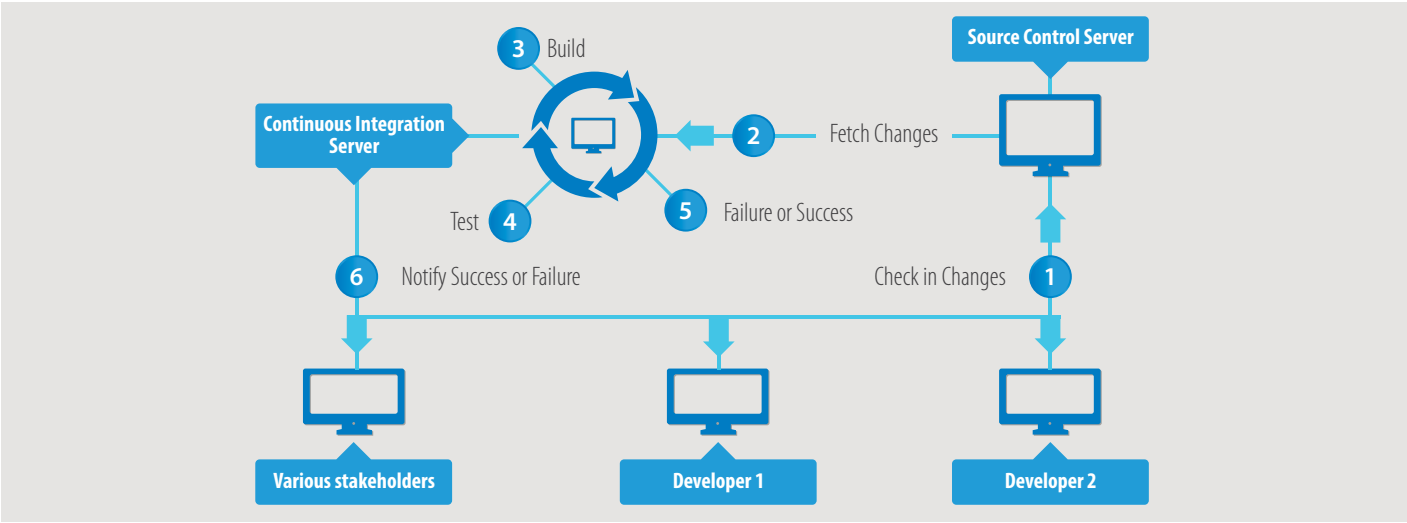
Section 3 discusses the challenges faced and the solution approaches we used.

Section 4 includes some of the best practices that can be adopted for CI

Section 5 enumerates some of the benefits obtained from CI

Section 6 shows the deployment topology for the CI setup

And we finally summarize our findings in **Section 7.**



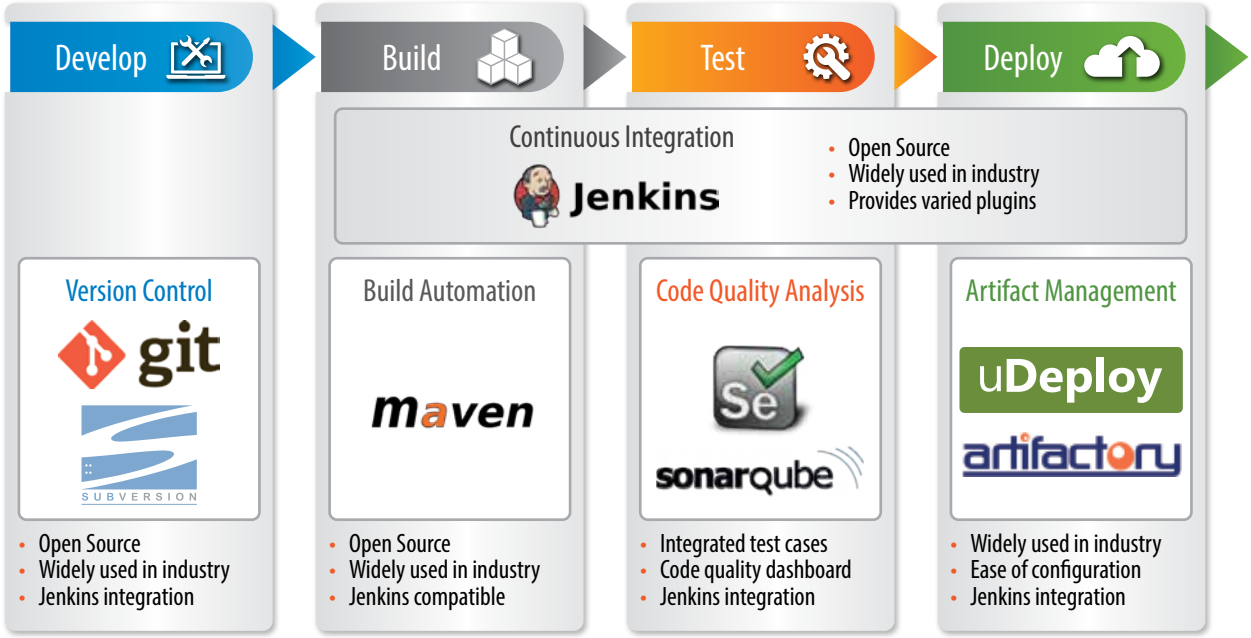
Continuous Integration Setup

The below diagram illustrates the end to end Continuous Integration (CI) setup which we have been following across projects. As seen below, the main actors include the Development team, the Source Control Server and the Continuous Integration server. Developers check-in the code into source control server which is integrated with CI server. For each build, CI server is configured to run the JUnit test cases, Selenium based functional test cases, code quality checks and provide notifications for any failure scenario which enables the development team to take immediate action. This continuous automation chain helps in reducing the overall defect density and thereby improving the code quality.

Tools Adoption

CI depends mostly on adopting the correct set of tools and their proper usage. The selection of tools is generally driven by various IT policies in the organization, existing technology landscape, current infrastructure setup, and other considerations. It is therefore recommended that every organization must do proper due diligence in evaluating different toolsets and choosing the appropriate ones suitable for their requirements.

The diagram below shows the toolsets (phase wise) which we have been using successfully across various projects for CI enablement. As seen, we have adopted Jenkins as the continuous integration platform but there are other CI platforms (Bamboo, TeamCity etc.) to choose from. The same case holds good for other toolsets also.



Challenges Faced

We faced a few challenges during our CI journey and some of the major challenges encountered are explained below.

Providing Granular Level Access In Jenkins

1

Providing individual level access was a redundant and time consuming task

Solution Approach

We came up with the role based access solution approach. This approach helped to provide access to roles rather than individual level

Regular Build Failures

2

Most common issue faced. Jobs were failing because of perm gen space issue. Memory consumption on slave nodes was high due to Jenkins job workspace and generated artifacts

Solution Approach

To solve this issue, cron scripts were created to clean up the job workspace and artifacts once a week, in order to release the memory

Builds Waiting for Node Executor

3

Due to node dependency, projects were waiting in the queue even if idle executors were available on other nodes. Builds were waiting for executor due to heavy build load

Solution Approach

Even Scheduler plugin was used to ensure that load distribution was happening evenly across all nodes

GIT Repository Integration

4

Issue while creating a trusted link between GIT and Jenkins server

Solution Approach

Trust was created between GIT and Jenkins using public and private keys of Jenkins server

Best Practices Followed

After having worked on multiple CI enablement projects we have condensed together the set of best practices which we have learnt/followed. Most of these practices have been standardized and published as CI guidelines for internal usage across teams. In this section we highlight some of the best practices which we follow and would like to share with others.

Role Based Access	Providing individual level access is always a cumbersome task and therefore it is recommended to provide Role-Based access
Regular cleanup of workspace & artifacts	Cron scripts were created to clean up the job workspace and artifacts once a week, in order to ,release the memory and thereby avoid memory leak issues
Even distribution of workload across nodes (Even Scheduler plugin)	To overcome the default behavior of Jenkins in terms of node picking job, Even Scheduling plugin was used which ensuresd even distribution of work load across nodes
Template based commons configuration	Template based configuration enabled centralization of common properties across projects

Benefits Realized

Following are some of the benefits which we have realized by CI enablement across projects.

Build Automation

100%

Builds were automated by using the Poll SCM feature of Jenkins. This not only enabled faster builds but also help to eliminate manual build effort

Code Stability

100%

Notification feature enabled development team to respond quickly to the build failure scenarios. This helped in quicker resolution of code issues and ultimately led to stable code base

Analytics

100%

Sonar dashboard provided centralized view of code quality issues and dashboards created leveraging Jenkins APIs, provided complete build information. This enabled project teams to take timely corrective steps

CD Enablement

100%

Workflow configuration on Jenkins enabled auto deployment of application artifacts (EAR/WAR files) on the required application server whenever a new snapshot version of the artifact was available

Faster Reeases

50%

CI enabled faster release cycles with stable code

Cost Savings

30%

Many projects cut down on the cost to maintain separate build teams after build automation. This resulted into significant cost savings for them

Improved Productivity

30%

CI-enablement helped increase development team productivity by relieving them from build related aspects with complete focus on their core development activities

Improved Code Quality

25%

CI-enablement helped increase development team productivity by relieving them from build related aspects with complete focus on their core development activities

Note: The above mentioned quantitative improvements are the average % improvements which have been observed across various projects.

Topology and Deployment view

Below is the high-level topology of the CI platform which we have been using. The key benefits of this topology are high availability and scalability. To achieve high scalability and distributed build system we have set up dedicated build nodes that run separately from the Jenkins master. This frees up resources for the master server to improve its scheduling performance.

Moreover, executing jobs on the master's executors can introduce a security issue. Any Jenkins's user with full permissions can play havoc with the system as they will have direct access to private information whose integrity and privacy could not be, thus, guaranteed. We leveraged the Jenkins supported "master/slave" mode, where the workload of building projects is delegated to multiple slave nodes.

If you notice we have enabled Jenkins master to only handle HTTP requests and manage the build environment. Actual execution of builds will be delegated to the slaves. With this configuration it is possible to horizontally scale the architecture.

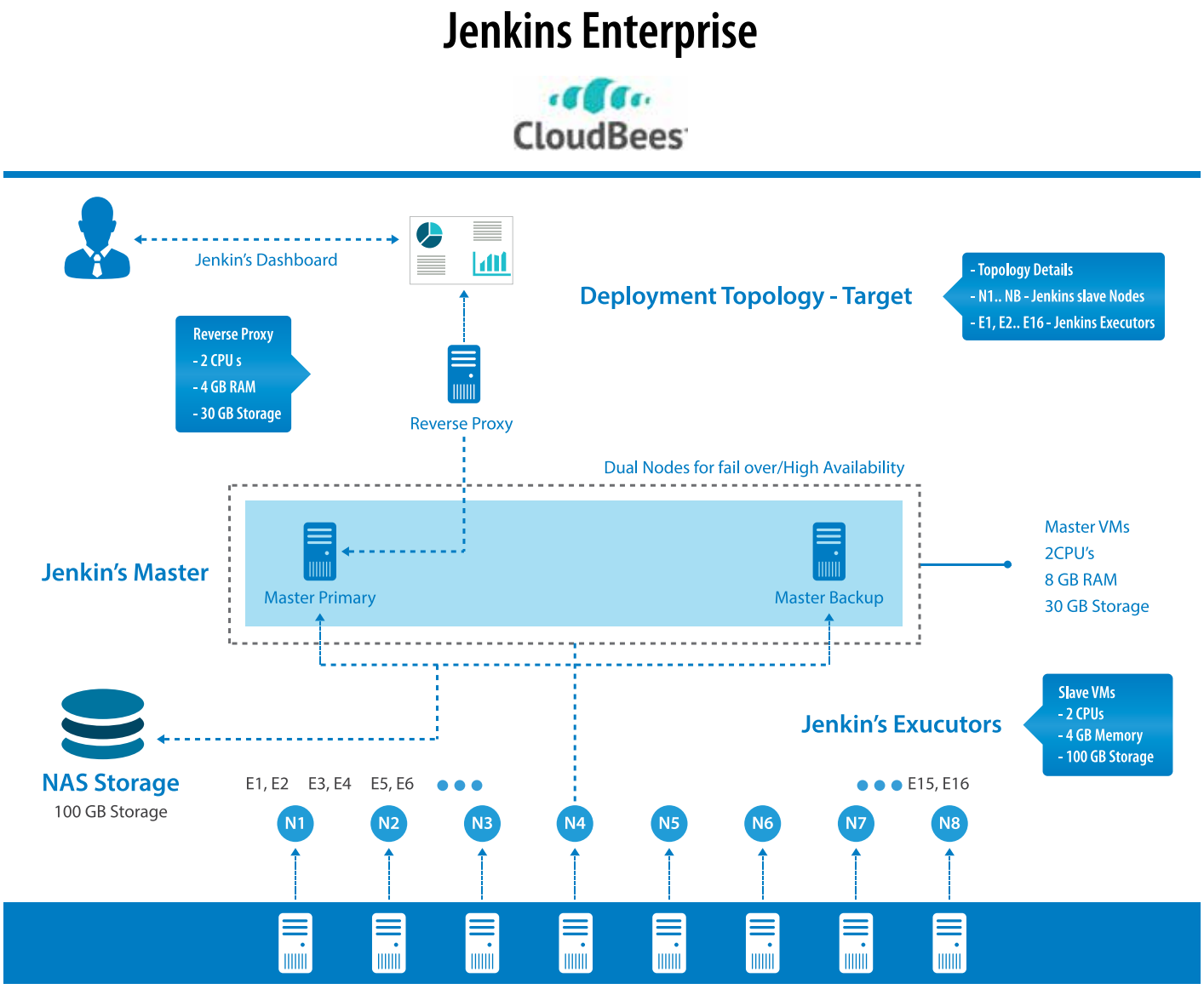


Fig 2. Continuous integration topology view

#	Component	Key Characteristics
1	Master and Backup Server	To avoid a single point of failure and support high availability we have the master slave configuration setup. In case the master goes down, a slave node can provide the continuity
2	NAS Storage	Responsible for sharing $JENKINS_HOME directory for all Jenkins Master VMs
3	Jenkins Slave Nodes	A slave node is a machine set up to offload build projects from the master node. Actual execution of the build happens on the slave node. The method for scheduling builds depends on the configuration given to a specific project. Some projects may be configured to only use a specific slave node while others can freely pick up a slave nodes from among the pool of slave nodes assigned to the master
4	Jenkins Executors	Responsible for concurrent builds on slave nodes. If a slave node consists of 4 executors , concurrently 4 jobs can be built on that node
5	Reverse Proxy Server	Responsible to ping all nodes in the cluster via a health check URL and uses their return codes to determine which node is the primary/active master

Conclusion

We believe that Continuous Integration is at the core of DevOps and needs proper planning. Since it is all about tools and their integration aspects, making appropriate choices based on various considerations is very important. Building good expertise on the tools involved, their integration aspects and other best practices, helps, avoid common pitfalls and helps speed up CI adoption.

In spite of knowing and following all the best practices, there will always be some or other challenges faced during CI adoption but the final benefits realized far outweigh these initial hiccups. We hope this article might have helped you in some way or the other and we would love to hear from you.

References

1. Gartner Says By 2016, DevOps Will Evolve From a Niche to a Mainstream Strategy Employed by 25 Percent of Global 2000 Organizations- http://www.gartner.com/newsroom/id/2999017
2. Distributed builds using Jenkins : https://wiki.jenkins-ci.org/display/JENKINS/Distributed+builds

About the Team

Amruta Kumbhar
Technology Lead

Amruta Kumbhar is a Product Technical Lead in Infosys. She has strong expertise on Java technologies and database technologies (Oracle, DB2, MongoDB). She also has very good expertise related to various CI aspects - build management using Maven and SonarQube plugin creation. She can be reached at Amruta_Kumbhar@infosys.com

Madhavi Shailaja
Technology Architect

Madhavi is a Technology Architect in Infosys. She has strong expertise on Java technologies. She also has very good expertise related to various CI aspects - build management using Maven and Ant. She can be reached at Madhavi_Katakam@infosys.com

Ravi Shankar Anupindi
Sr. Technology Architect

Ravi is a Senior Technology Architect in Infosys. Ravi has strong expertise on Java technologies and leads the Performance Engineering COE in one of the business verticals. His areas of interest include exploring latest technologies and looking at ways to adopt them to derive significant business benefits. He has been actively involved in DevOps initiatives for various clients. He can be reached at Ravi_Anupindi@infosys.com

For more information, contact askus@infosys.com

Stay Connected